Punkt 1: **Struktura klas (Modelowanie obiektowe)**

- **1. Klient**
  - **Rola:** reprezentuje dane klienta – model danych (imię, nazwisko, telefon, email)
  - **Plik:** Klient.cs
  - **Przestrzeń nazw:** ObslugaHotelu.Models
  - **Zastosowanie:** klasa z właściwościami, konstruktorem, metodami ToString() i FromString()
- **2. KlientManager**
  - **Rola:** logika biznesowa zarządzania klientami (dodawanie, pobieranie)
  - **Plik:** KlientManager.cs
  - **Przestrzeń nazw:** ObslugaHotelu.Managers
  - **Zastosowanie:** klasa z metodami statycznymi do operacji na danych klientów
- **3. WolnePokojeForm : Form**
  - **Rola:** formularz pokazujący wolne pokoje w tabeli
  - **Plik:** WolnePokojeForm.cs
  - **Przestrzeń nazw:** ObslugaHotelu.Forms
  - **Zastosowanie:** dziedziczy po Form, tworzy UI z DataGridView, ma prywatne metody (WczytajWolnePokoje)
- **4. RezerwujPokojForm : Form**
  - **Rola:** formularz służący do rezerwacji pokoju
  - **Plik:** RezerwujPokojForm.cs
  - **Przestrzeń nazw:** ObslugaHotelu.Forms
  - **Zastosowanie:** obsługuje logikę rezerwacji, operuje na danych z plików, dziedziczy po Form
- **5. MenuGlowneForm : Form**
  - **Rola:** główne menu aplikacji – punkt startowy GUI
  - **Plik:** MenuGlowneForm.cs
  - **Przestrzeń nazw:** ObslugaHotelu.Forms

- **Zastosowanie:** inicjuje inne formularze jako obiekty, np. new WolnePokojeForm().ShowDialog();

◆ **6. DodajKlientaForm : Form**

- **Rola:** formularz do wprowadzania danych nowego klienta
- **Plik:** DodajKlientaForm.cs
- **Przestrzeń nazw:** ObslugaHotelu.Forms
- **Zastosowanie:** działa na obiekcie Klient, współpracuje z KlientManager

◆ **7. DodajRezerwacjeForm : Form**

- **Rola:** formularz umożliwiający dodanie rezerwacji
- **Plik:** DodajRezerwacjeForm.cs
- **Przestrzeń nazw:** ObslugaHotelu.Forms
- **Zastosowanie:** operuje na danych pokoi, klientów i dat

◆ **8. DodajPokojForm : Form**

- **Rola:** formularz dodający nowe pokoje do systemu
- **Plik:** DodajPokojForm.cs
- **Przestrzeń nazw:** ObslugaHotelu.Forms
- **Zastosowanie:** logika zapisu danych pokoju do pliku

◆ **9. ListaKlientowForm : Form**

- **Rola:** pokazuje listę klientów
- **Plik:** ListaKlientowForm.cs
- **Przestrzeń nazw:** ObslugaHotelu.Forms
- **Zastosowanie:** odczyt danych z pliku, generowanie widoku tabeli

◆ **10. ListaRezerwacjiForm : Form**

- **Rola:** wyświetla wszystkie rezerwacje
- **Plik:** ListaRezerwacjiForm.cs
- **Przestrzeń nazw:** ObslugaHotelu.Forms
- **Zastosowanie:** pobiera dane z pliku rezerwacje.txt, przetwarza i prezentuje

Podsumowanie:

Klasa	Typ	Lokalizacja	Rola główna
Klient	Model danych	Models/Klient.cs	Reprezentuje klienta
KlientManager	Logika	Managers/KlientManager.cs	Zarządza operacjami na klientach
WolnePokojeForm	Formularz	Forms/WolnePokojeForm.cs	Pokazuje dostępne pokoje
RezerwujPokojForm	Formularz	Forms/WolnePokojeForm.cs	Rezerwowanie pokoju
MenuGlowneForm	Formularz	Forms/WolnePokojeForm.cs	Główne menu
DodajKlientaForm	Formularz	Forms/WolnePokojeForm.cs	Dodawanie nowego klienta
DodajRezerwacjeForm	Formularz	Forms/WolnePokojeForm.cs	Tworzenie rezerwacji
DodajPokojForm	Formularz	Forms/WolnePokojeForm.cs	Dodawanie nowego pokoju
ListaKlientowForm	Formularz	Forms/WolnePokojeForm.cs	Lista klientów
ListaRezerwacjiForm	Formularz	Forms/WolnePokojeForm.cs	Lista rezerwacji

2: **Dziedziczenie** – wszystkie przykłady z lokalizacją

W aplikacji dziedziczenie występuję w kontekście interfejsu użytkownika. Formularze dziedziczą po klasie bazowej Form z biblioteki System.Windows.Forms. Jako przykład

Dziedziczenia możemy podać tak naprawdę każdą klasę z folderu Form, przykład : public class WolnePokojeForm **: Form**

3: **Polimorfizm** – wszystkie przykłady z lokalizacją w kodzie

Polimorfizm pozwala różnym klasom na wspólne interfejsy/metody, ale z różną implementacją lub zachowaniem.

3.1. **Polimorfizm przez dziedziczenie (działanie wspólnych metod Form)**

Wszystkie klasy dziedziczące po System.Windows.Forms.Form mogą być używane zamiennie jako okna, mimo że każda z nich ma inny kod i wygląd.

Przykłady

Klasa	**Metoda wykorzystywana polimorficznie**	**Lokalizacja**
WolnePokojeForm	Show()	Forms/WolnePokojeForm.cs
RezerwujPokojForm	Show()	Forms/RezerwujPokojForm.cs
DodajKlientaForm	Show()	Forms/RezerwujPokojForm.cs
DodajRezerwacjeForm	Show()	Forms/DodajRezerwacjeForm.cs

Każda z tych klas wywołuje Show(), ale sposób działania zależy od konkretnej klasy – to przykład **polimorfizmu dynamicznego**.

3.2. **Polimorfizm przez przesłanianie metod**

W klasie Klient, przesłonięto metodę ToString(), dziedziczoną z klasy object. Dzięki temu obiekt Klient może być zamieniany na tekst w określony sposób.

Przykład: public override string ToString()

{

return $"{Imie},{Nazwisko},{Telefon},{Email}";

}

Umożliwia zamianę obiektu klienta na tekst w określonym formacie (np. przy zapisie do pliku), różnie niż domyślna implementacja klasy object.

Podsumowanie:

Typ polimorfizmu	**Klasa lub metoda**	**Lokalizacja**	**Opis**
Przesłonięcie ToString()	Klient	Models/Klient.cs	Specjalna wersja tekstu reprezentująca klienta
Metoda FromString()	Klient	Models/Klient.cs	Tworzy obiekt Klient z tekstu
ShowDialog() w formularzach	Wszystkie klasy pochodne od Form	Forms/*.cs	Różne formularze – jedna metoda, różne działanie

4: **Hermetyzacja (enkapsulacja)** – wszystkie przykłady z lokalizacją w kodzie

Hermetyzacja polega na ukrywaniu wewnętrznych szczegółów działania klasy i udostępnianiu tylko kontrolowanych interfejsów do pracy z danymi. Chroni dane przed niepożądanym dostępem lub modyfikacją.

4. 1. **Pola i właściwości klasy Klient**

Klasa Klient udostępnia dane tylko przez właściwości (get/set), co pozwala na pełną kontrolę nad dostępem do danych.

Hermetyzacja: dane klienta nie są przechowywane w publicznych polach, lecz we właściwościach z możliwością ewentualnego dodania walidacji lub logiki później.

4.2. **Prywatne pola i logika pomocnicza w KlientManager**

private static readonly string FilePath = … - Hermetyzacja: ścieżka pliku oraz szczegóły implementacji odczytu/zapisu są ukryte przed resztą aplikacji. Dostęp do danych odbywa się wyłącznie przez publiczne metody:

public static void DodajKlienta(Klient klient)

public static List<Klient> PobierzWszystkich()

4.3 Prywatne kontrolki i metody w formularzach

private DataGridView dgvWolnePokoje;

private void WczytajWolnePokoje() { … }

Hermetyzacja: formularz posiada prywatne kontrolki (np. DataGridView), do których nie można się dostać z zewnątrz – użytkownik formularza widzi tylko okno i interfejs, ale nie ma dostępu do jego wewnętrznej logiki.

To samo dotyczy wszystkich innych formularzy, np.:

- RezerwujPokojForm.cs
- DodajKlientaForm.cs
- DodajRezerwacjeForm.cs

Element chroniony	**Gdzie ?**	Typ hermetyzacji
Dane klienta	Klient.cs	Udostępnione przez właściwości get/set
Ścieżka pliku	KlientManager.cs	Ukryta jako private static readonly

Metody zarządzające danymi	KlientManager.cs	Publiczne API – ukryta logika
Dane GUI (kontrolki)	Np. WolnePokojeForm.cs	Prywatne pola i metody w formularzach
Logika wczytywania danych	Np. WolnePokojeForm.cs	Ukryta przed użytkownikiem GUI